1. Determination of the LPS
2. Calculation of the Variables R_{LPS} and R_{MPS}:

```
R_LPS = R x P_LPS
R_MPS = R - R_LPS
```

3. Calculation of the new partial interval:

```
if (bit = LPS) then
        L ← L + R_MPS
        R ← R_LPS
else
        R ← R_MPS
```

4. Updating the probability estimation P_{LPS}
5. Outputting the bits and renormalizing R

FIG. 1

1. Determination of the LPS
2. Quantization of R:

```
q_index = Qtab[R>>q]
```

3. Determination of R_{LPS} and R_{MPS}:

```
R_LPS = Rtab[q_index,p_state]
R_MPS = R - R_LPS
```

4. Calculation of the new partial interval:

```
if (bit = LPS) then
        L ← L + R_MPS
        R ← R_LPS
        p_state ← Next_State_LPS[p_state]
else
        R ← R_MPS
        p_state ← Next_State_MPS[p_state]
```

FIG. 2

1. Determination of the LPS
2. Quantization of R:

```
q_index = Qtab[R>>q]
```

3. Determination of R_{LPS} and R_{MPS}:

```
R_LPS = Rtab[q_index,p_state]
R_MPS = R - R_LPS
```

4. Determination of bit, depending on the position of the partial interval:

```
if (V ≥ R_MPS) then
      bit ← LPS
      V ← V - R_MPS
      R ← R_LPS
      p_state ← Next_State_LPS[p_state]
else
      bit ← MPS
      R ← R_MPS
      p_state ← Next_State_MPS[p_state]
```

5. Renormalization of R, reading out a bit and updating V

FIG. 3

Encoder:

1. Calculating the new partial interval:

```
R ← R >> 1
if (bit = 1) then
      L ← L + R
```

2. Outputting bits and renormalizing R

Decoder:

1. Determination of bit, depending on the position of the partial interval:

```
if (V ≥ R) then
      bit ← 1
      V ← V - R
else
      bit ← 0
```

2. Reading out a bit, renormalizing R and updating V

FIG. 4

Encoder:

1. Calculating the new partial interval:

```
L ← L << 1
if (bit = 1) then
      L ← L + R
```

2. Outputting a bit and renormalizing using doubled determination threshold values (without doubling R and L)

Decoder:

1. Reading out a bit and updating V
2. Determination of bit depending on the position of the partial interval:

```
if (V ≥ R) then
      bit ← 1
      V ← V - R
else
```

FIG. 5

```
1. preState = min(max( 1,((m * SliceQP )>>4)+n),126)
2. if (preState <= 63) then
         p_state = 63 - preState
         valMPS = 0
   else
         p_state = preState - 64
         valMPS = 1
```

FIG. 6